LE STATISTICIEN ET L'INFORMATIQUE

Par : Jean-Pierre GRANDJEAN

L'informatique a investi la statistique. L'évolution est banale, générale, définitive. Elle n'est pas encore terminée. Si en un peu plus de dix ans, le fait a été acquis, il reste encore à subir l'intrusion des mini ou micro-informatiques, conséquence d'un développement technologique imprévisible, accompagné d'une chute étonnante des coûts du matériel.

L'INSEE a à peu près réussi à suivre le mouvement. Son informatique a cru et embelli. Des structures se sont mises en place : centres informatiques, procédures de gestion, plus ou moins difficilement, comme cela se conçoit au sein d'une organisation de ce type. Des structures lourdes, dont l'adaptation aux évolutions techniques n'est pas toujours aisée, surtout quand on voit poindre, enfin, une informatique peut-être plus accessible, plus "conviviale", plus proche des utilisateurs. Les réflexions récentes menées à l'occasion de la programmation du moyen terme s'en font l'écho, où sont reconnues pertinentes, des notions telles que : informatique intégrée, libre-service, etc ... (il ne s'agit pas ici de s'étendre sur les diverses significations possibles de ces termes).

Le progrès technique qui en 1980 fait tenir dans une boite d'allumettes l'unité centrale qui nécessitait plusieurs mètres cubes en 1950, ne s'est pas accompagnée, tant s'en faut, d'une semblable amélioration de l'accès à l'ordinateur. Pourquoi ? Les raisons sont multiples et interdépendantes, et essentiellement relatives au fait que pour utiliser un ordinateur, il est nécessaire de disposer de programmes (le logiciel ou "software" en anglais).

La réflexion sur les méthodes et les outils permettant d'écrire de bons programmes a été menée parallèlement aux progrès de la quincaillerie, pas avant malheureusement. Il a fallu souvent, dans les premiers temps, se débrouiller avec "les moyens du bord", créer rapidement des outils permettant quand même de s'éviter l'écriture de programmes en langage binaire (ou plutôt en hexadécimal ou en octal), sans disposer de support théorique et de l'expérience nécessaires.

Ces outils imparfaits ont été très vite très employés, et se sont avérés très difficiles à remplacer par d'autres, meilleurs à tous points de vue, mais qui avaient le handicap d'arriver trop tard. Un apprentissage prend du temps, toute modification du stock énorme de programmes existants coûte cher. L'inertie et la routine des informaticiens sont importantes. Mais cela se comprend, car on s'est aperçu, plus ou moins vite, et ça a été parfois dur à admettre, qu'écrire de bons programmes qui "marchent" est un exercice délicat.

Si les erreurs d'analyse sont très souvent rédhibitoires, les erreurs de programmation sont bien ennuyeuses. Il n'est guère possible de demander aux programmeurs d'être productifs et de changer d'habitude tous les six mois ; par exemple utiliser un nouveau langage de programmation, cela signifie l'apprendre, savoir comment le mettre en oeuvre, connaître ses finesses ou les finesses du compilateur, qui ne sont pas toujours évidentes.

Ceci explique pourquoi, en 1981, la compagnie IBM ose encore imposer à la majeure partie de ses utilisateurs l'emploi du barbare JCL, et pourquoi un langage aussi frustre que le FORTRAN n'a pas encore été supplanté. Il n'empêche que des progrès importants ont quand même été faits. Dans le domaine des langages de programmation, la décennie 1980 verra peut-être le renversement des anciennes idoles et par exemple l'installation d'ADA, langage puissant, créé à l'initiative du Ministère de la Défense des Etats-Unis.

Les terminaux légers, écrans ou machines à écrire, doivent se généraliser. Si les outils logiciels qui les accompagnent (éditeurs de texte, etc ...) sont de bonne qualité, des gains importants de productivité sont possibles.

Les logiciels généraux s'améliorent, profitant des acquis antérieurs, outils qui permettent à l'utilisateur non informaticien de formuler ses demandes dans un langage non purement informatique, et plus proche, dans une certaine mesure, de celui de sa discipline.

Dans ce paysage mouvant, l'utilisateur est quand même fortement balloté. Et l'on voit des statisticiens qui nient l'informatique, des statisticiens craintifs, qui font toujours la même chose par peur des résultats curieux que peut produire le déplacement d'une virgule, des statisticiens audacieux qui attaquent un peu à la légère des problèmes fort ardus, des statisticiens conquérants qui veulent réinventer la poudre, et aussi des statisticiens sereins, qui ont pris la mesure de l'outil et l'utilisent à peu près à bon escient.

Dans les paragraphes qui viennent, nous voudrions décrire, plus ou moins précisément, quelques-uns des problèmes qui se posent au statisticien dans l'utilisation autonome de l'informatique. Remarquons par avance qu'ils ne sont pas identiques pour celui qui doit traiter une enquête de deux cent mille unités et pour celui qui s'intéresse à une série temporelle de quatre vingt quatre points Ils ont quand même un large fondement commun.

Le propos est plutôt orienté vers les études statistiques, de la tabulation à la modélisation. On ne traitera pas de problèmes de pure gestion (tels que les abordent par exemple les systèmes de gestion de bases de données), encore que le statisticien, pour ses analyses, soit toujours confronté à des problèmes de cet ordre.

Le statisticien de l'INSEE, pour obtenir des chiffres, peut procéder de diverses façons. Il peut tout sous-traiter à un centre informatique, il peut écrire ses propres programmes de A jusqu'à Z, utiliser des logiciels généraux ou des programmes déjà écrits, et constituer des chaînes de traitements dont les différents programmes communiquent au moyen de fichiers intermédiaires. Toute solution qui fournit des résultats corrects est bonne, la meilleure étant, nous semble-t-il, celle qui limite le temps passé.

I. L'ECRITURE DE PROGRAMMES SPECIFIQUES

Elle ne doit s'envisager qu'en dernier ressort, lorsque toute autre solution, à base de logiciels généraux ou de programmes préexistants, a été jugée inadéquate. C'est une entreprise périlleuse, qui peut prendre un temps déraisonnable en regard de l'objectif visé, et fournir des résultats imparfaits. Il faut insister sur l'aspect temporel. Des études de productivité faites sur de nombreux projets informatiques, aux Etats-Unis, ont montré que la production moyenne d'un programmeur s'élève à quelques lignes (de cinq à trente) de programme correctes par jour. Il ne faut pas oublier qu'avant d'utiliser un programme, il faut corriger les erreurs qu'il contient presque toujours, que, lorsqu'on l'utilise, on découvre de nouvelles erreurs, qu'il faut également corriger. Tout cela prend énormément de temps, a fortiori pour quelqu'un dont ce n'est pas l'activité principale.

L'écriture de programmes est classiquement décomposée en deux phases, analyse et programmation proprement dites.

1- L'analyse

a) Cette étape primordiale est trop souvent négligée. DESCARTES a fort bien défini de quoi il s'agit. Examiner et définir le problème, le découper en sous-problèmes, en bref définir l'algorithme de résolution. Il existe des bibliothèques d'ouvrages sur la question. Signalons simplement que cette réflexion préalable permet bien souvent de gagner du temps par la suite, d'abord en évitant de se lancer dans des constructions absurdes, ensuite lors de la programmation, en permettant la détection de sous-programmes ou modules classiques, qui ont déjà été écrits et qu'il suffit de recopier, par exemple : inversion ou diagonalisation de matrice, tri de vecteur, etc ...

Il est idiot de vouloir à toute force réinventer la poudre. Ecrire son propre programme d'inversion de matrice, c'est perdre son temps, puisque d'autres l'ont fait avant ; c'est courir le risque d'obtenir des résultats faux : le problème n'est pas toujours aussi trivial qu'il y paraît au premier abord et la solution adoptée peut s'avérer d'un coût informatique trop élevé si la matrice est de

rang important.

b) Les bibliothèques de sous-programmes

Les bibliothèques sur support magnétique

Certains organismes, publics ou privés, n'ont pas manqué de rassembler des sous-programmes classiques, écrits par des gens compétents. Certains de ces en sembles sont très diffusés de par le monde, ce qui fait qu'un grand nombre de leurs défauts ont pu être détectés et corrigés. Certains de ces projets ont d'ailleurs été l'occasion d'une réflexion fondamentale sur des sujets importants tels que la précision, ou la portabilité dont on parlera plus tard. Sur l'ordinateur IBM du CNI P1 sont disponibles les bibliothèques IMSL et HARWELL, dont les qualités sont largement reconnues, et qui contiennent grand nombre d'algorithmes classiques. Cette solution est de loin la plus efficace, car ces sous-programmes sont directement utilisables depuis des programmes FORTRAN.

Les algorithmes publiés

Un certain nombre de revues publient des algorithmes. Citons :

- Applied Statistics ;
- Communications in Statistics (partie B) ;
- Transactions on Mathematical Software ;
- Computer Journal.

Les algorithmes sont normalement vérifiés par l'éditeur, de temps en temps certifiés par des utilisateurs. Ils posent parfois certains problèmes, de langage, s'ils sont écrits en ALGOL ou en ALGOL 68, de portabilité, s'ils sont écrits pour telle machine, qui n'est pas celle dont vous disposez. Souvent, une simple adaptation est préférable à une réécriture complète, même s'il ne faut pas négliger certaines difficultés techniques.

c) En fin de compte, les algorithmes publiés (ceux d'IMSL, de HARWELL, du NAG, ou ceux des revues sérieuses) offrent des garanties intéressantes : algorithme adéquat, performant, donnant des résultats justes, ou précisant la ou les

raisons d'une absence ou d'une imprécision éventuelle des résultats : par exemple matrice mal conditionnée, ou strictement non inversible. Leur documentation est relativement normalisée. Elle décrit leur mode d'emploi, leurs conditions d'emploi (matrice quelconque ou symétrique, symétrique quelconque ou bande tridiagonale, etc .).

2- La programmation

Même lorsqu'on a réussi à isoler des modules déjà écrits, il faut encore écrire quelques lignes de programme. Se posent un certain nombre de problèmes :

a) Le choix de langage de programmation

Les langages classiques offerts par exemple à l'utilisateur de l'ordinateur IBM du CNI P1 sont le FORTRAN, le COBOL, et le PL1.

Le COBOL n'attire pas les foules, étant donné son extrême verbosité. Le FORTRAN est relativement simple, souvent enseigné comme langage de base. Il est parfois insuffisant pour certaines applications sur fichiers complexes, et faible pour le traitement de l'alphanumérique. Un gros avantage est que les biblio thèques de sous-programmes sont à 99 % écrites en FORTRAN. Sur matériel IBM, un point noir est le compilateur, qui, s'il est très efficace, aide très mal l'utilisateur à détecter et à corriger ses erreurs.

Le PL1 est fort prisé de ses pratiquants, puisqu'il présente à la fois les caractéristiques d'un langage de gestion tel que le COBOL et celles d'un langage de type scientifique tel que le FORTRAN. Le compilateur PL1 d'IBM permet une recherche efficace des erreurs. Il est très possible d'appeler depuis un programme PL1 des sous-programmes FORTRAN, ce qui n'exclut pas le recours aux bibliothèques de sous-programmes. C'est encore plus facile avec le compilateur IBM le plus récent, qui n'est pas installé au CNI P1.

Signalons que le langage Pascal, à notre sens, n'est pas actuellement susceptible de remplacer les langages précédents, du fait de ses insuffisances, dans le domaine des lectures ou des écritures de fichiers principalement.

Le BASIC fait la joie des utilisateurs de micro-ordinateurs. Le BASIC de base est moins intéressant que le FORTRAN. Chaque constructeur est très fier de ses extensions, qui par exemple sur le matériel WANG font du BASIC un langage utilisable. Les interpréteurs de BASIC permettent une exécution pas à pas des programmes, ce qui peut aider notablement dans la tâche de correction des erreurs. Ce genre de produit existe évidemment sur les gros ordinateurs, mais n'est de toute façon pas très utilisable sans terminal léger.

Un mot d'APL. APL est un langage très concis, très symbolique, utilisant des concepts très mathématisés. C'est un langage d'ingénieur. Il serait utile de réexaminer à la lumière des versions actuelles du langage (et de ses interpréteurs) ce qui constituait un handicap insurmontable pour de nombreux statisticiens de l'INSEE : l'insuffisance des lectures et écritures sur fichier magnétique. Sur matériel IBM, ce problème semble résolu, la portabilité (Cf plus loin) des applications utilisant cette solution étant parfaitement nulle. Un handicap certain d'APL demeure qu'il est très coûteux de l'employer pour des algorithmes complexes.

b) Exactitude des résultats obtenus

Ce sujet important mérite qu'on s'y arrête.

Il y a d'abord, bien sûr, la transcription erronée de formules justes ou fausses. Il n'est pas toujours aisé de s'apercevoir de telles erreurs. Plus fondamental est le point suivant. Dans un ordinateur, les nombres sont représentés à l'aide d'un nombre fini de positions binaires. Ce fait est à l'origine de nombreuses erreurs ou inexactitudes, et nécessite donc d'être commenté.

En calcul scientifique, les arithmétiques couramment employées sur les ordinateurs sont l'entière et la "flottante" (ou réelle).

L'arithmétique entière

Les nombres entiers sont représentés avec M bits (positions binaires) dont un de signe ; donc tout nombre entier compris entre -N et +N (N = f (M)) est représenté de manière exacte et, du moment que les résultats des opérations faites appartiennent à l'intervalle (-N, +N), tout marche bien. Ca se corse lorsqu'on sort des bornes. Par exemple, sur un ordinateur IBM, si les bornes sont

$(-2^{31}, 2^{31}-1)$ pour des entiers de 32 bits, on peut vérifier que l'opération suivante $(2^{31}-1)+1$ fournit, programmée en FORTRAN, le résultat inattendu suivant : -2^{31} ; et ceci sans prévenir l'utilisateur par quelque message que ce soit. Gare donc, lorsque vous ajoutez de grands nombres positifs, en arithmétique entière.

L'arithmétique "flottante" (ou réelle)

Pour les calculs sur nombres réels, les différents constructeurs d'ordinateurs ont inventé de nombreuses arithmétiques dites "flottantes", dans lesquelles les nombres sont représentés sous la forme exposant + mantisse.

Ces arithmétiques diffèrent par la base utilisée (16 sur IBM), le nombre des bits réservés à l'exposant et à la mantisse respectivement, ceci étant fonction du nombre de bits total accordé à un nombre, le traitement du signe, etc .

Sur un ordinateur IBM, en arithmétique flottante standard ou simple précision, un nombre est représenté sur 32 bits. Le plus grand nombre réel positif exactement représentable est 16^{63} $(1-16^{-6})$. Le plus petit réel positif exactement représentable est 16^{-65}. La précision moyenne des nombres représentables est 6,3. En gros, on ne peut donc pas espérer de résultats justes à plus de 6 décimales. Ceci est faible. IBM offre une arithmétique flottante double précision guère plus coûteuse, dans laquelle les nombres sont représentés sur 64 bits. La précision obtenue est de l'ordre de 16 décimales.

Il existe pour mémoire une arithmétique flottante quadruple précision, dans laquelle les nombres sont représentés sur 128 bits, la précision étant alors d'environ 34, mais dont le coût d'usage prohibitif doit limiter l'emploi à des cas exceptionnels.

Bien évidemment la description de ces arithmétiques ne se limite pas à l'énoncé des faits ci-dessus rappelés. Elle comporte également les algorithmes employés pour les diverses opérations, addition, soustraction, multiplication, division, etc .

Donnons quelques exemples, limités à l'addition, des conséquences importantes que l'emploi de telles arithmétiques entraîne sur la précision des

résultats, les termes techniques employés pour nommer ces phénomènes étant : erreurs d'arrondi et propagation des erreurs d'arrondi. Ces exemples sont opératoires sur IBM et CII.

1. $10^7 + 1 = 10^7$ en arithmétique flottante simple précision.

2. Toujours en arithmétique simple précision, supposons qu'on veuille ajouter 100 000 fois le nombre 1 000.

Un programme FORTRAN qui effectue ce calcul peut s'écrire :

```
X = 0
DØ 10 I = 1,100000
X = X + 1000
10  CØNTINUE.
```

Le résultat qu'on obtient n'est pas 10^8, comme on pourrait s'y attendre, mais 99334224.

A partir d'un certain moment, on atteint la limite de précision. Au lieu d'ajouter 1000, l'ordinateur ajoute en fait 992 = 16 X 62. Vient même un moment où il ajoute 768 = 16^2 X 3.

Imaginez qu'en réalité vous calculiez la moyenne pour 100 000 individus de nombres voisins de 1 000. Si vous utilisez l'algorithme suivant :

Calculer $S = \sum_{i=1}^{100000} X_i$

Calculer $M = \frac{S}{100000}$

le résultat pourra être relativement inexact.

3. Un autre exemple du même style : soit à calculer la variance des 3 nombres 9 001, 9 002, 9 003 par l'algorithme classique. Vous calculez successivement :

$$S_2 = \sum_{i=1}^{3} X_i^2 \quad ,$$

$$S = \sum_{i=1}^{3} X_i \quad ,$$

$$M = S/3 \quad ,$$

$$NUM = S_2 - 3M^2 \, .$$

et vous obtenez comme résultat 0, alors que vous espériez 2.

Pour des calculs aussi élémentaires que des calculs de moyennes et de variances, il faut donc bien souvent prendre des précautions telles que :

- utiliser la double précision ;
- utiliser des algorithmes progressifs.

Pour une moyenne : $M_1 = X_1$

$$M_k = M_{k-1} + \frac{X_k - M_{k-1}}{k}$$

$$\bar{X} = M_n$$

Pour un calcul de variance, le meilleur algorithme possible est :

I calculer la moyenne (correctement)
2 calculer la somme des carrés des écarts à la moyenne.

Cet algorithme est souvent coûteux, puisqu'il peut, par exemple, nécessiter deux lectures de fichier.

On peut utiliser à la place l'algorithme suivant :

$$T_1 = 0$$

$$T_k = T_{k-1} + (k-1) \times (X_k - M_{k-1}) \times \frac{X_k - M_{k-1}}{k}$$

$$V = \frac{T_n}{n-1}$$

La diffusion des erreurs d'arrondi est un phénomène universel, auquel ne peut échapper la moindre inversion de matrice, la plus petite diagonalisation. Les bons algorithmes, dits stables, sont ceux qui minimisent cette propagation. Leur recherche et leur étude ont été un sujet de préoccupation important des analystes numériciens, depuis le développement de l'informatique. C'est une des raisons majeures qui font militer pour l'emploi de sous-programmes de bibliothèques, qui reprennent normalement les "bons" algorithmes.

c) Performances

L'aspect "performances" ne doit pas être considéré comme fondamental en première analyse. Pour beaucoup de programmeurs, performance est synonyme d'astuce de programmation, bien souvent de programme illisible, pour l'auteur y compris, après un certain temps, car il ne réussit plus à comprendre ce qu'il a écrit Cependant, tel programme, dont la consommation est raisonnable pour un problème de taille 100, deviendra un gouffre pour un problème de taille 100 000, parce que la consommation de l'algorithme est en n^2, n^3 voire n^4.

Que faire ?

1. Sans modifier l'algorithme, il faut utiliser certaines caractéristiques de l'ordinateur, qui permettent d'aller plus vite.

- Choisir les bonnes méthodes de lecture - écriture sur fichier
- Mettre en oeuvre les options d'optimisation des compilateurs (par exemple FORTRAN H Extended, Option Optimize de COBOL, PL1 Optimizing).

2. Modifier l'algorithme. Ceci est plus ou moins délicat.

- De façon conceptuellement évidente, il est par exemple facile de tenir compte des symétries d'un problème pour réduire énormément la quantité de calculs.

- D'un autre côté, une modification insignifiante, mais non triviale, peut transformer considérablement le comportement d'un algorithme. Un exemple célèbre est le tri en mémoire pour lequel, à partir d'algorithmes simples, dont la consommation est en n^2 ou en n^3, on peut par des finesses peu évidentes obtenir des algorithmes en n Logn, borne inférieure. L'algorithme a bien sûr été l'occasion de recherches et de publications nombreuses.

d) Portabilité

Quelques idées sur ce point qui sera repris plus tard. L'utilisateur peut être confronté à ce problème, quand par exemple il a écrit et fait fonctionner un programme sur un IRIS 80 et veut le transférer sur un IBM. Il s'aperçoit avec désappointement que ce programme ne veut plus tourner. Les ordinateurs sont nombreux et l'informatique n'est pas outrageusement standardisée, c'est le moins qu'on puisse dire. Si l'on considère les langages de programmation pour lesquels il existe une norme internationale, par exemple FORTRAN, COBOL, PL1, on s'aperçoit que les constructeurs ne s'entendent pas toujours sur son interprétation et qu'ils rajoutent souvent au langage de base des extensions plus ou moins commodes, plus ou moins justifiables. Dès qu'un programmeur utilise ces extensions, il s'expose à affronter les difficultés décrites plus haut. De façon plus subtile, quand on transfère un programme d'un ordinateur à un autre, il faut tenir compte des nombres de bits sur lesquels sont représentés les nombres pour chacun des ordinateurs concernés, et par exemple opérer la transformation des variables simple précision en variables double précision lorsqu'on passe d'un ordinateur CDC (mots de 60 bits) à un ordinateur IBM (mots de 32 bits) si l'on ne veut pas perdre trop de précision dans les résultats.

e) Documentation

Un programme non documenté est un programme non réutilisable.

Cette proposition est à peine excessive. Comprendre ce que calcule un programme non documenté prend énormément de temps. Si le programme est long et com plexe, ça peut s'avérer impossible. Une documentation de programme est plus ou moins complète.

- Le mode d'emploi précis et complet est indispensable dans tous les cas.

- Des commentaires parlants,insérés dans le texte du programme, sont quasiment nécessaires.

- Le dossier d'analyse - programmation complet est fort utile lorsqu'on veut transmettre un programme à quelqu'un d'autre.

Documenter un programme prend moins de temps que le réécrire.

En conclusion, répétons, au risque de lasser, que la bonne programmation est un art difficile.

En s'y lançant sans précautions particulières, on s'expose à de nombreux accidents imprévus, et on risque surtout d'y consacrer beaucoup plus de temps que prévu et raisonnable.

Pour améliorer cette situation, malheureusement trop fréquemment rencontrée, on peut espérer :

- une meilleure formation. Cette formation de base doit donner des idées claires sur la façon dont fonctionne un ordinateur, sur la façon dont sont stockées les informations dans un ordinateur ., sur la notion de fichier, sur les algorithmes de base. Elle doit également inculquer des règles de bonne programmation, à base de bon sens, et sans dogmatisme.

- un emploi plus systématique des bibliothèques de sous-programmes.

- le triomphe, sur les ordinateurs, de langages de programmation plus puissants dont les compilateurs assistent beaucoup plus l'utilisateur dans la tâche pénible de détection et correction de ses erreurs.

II. LES LOGICIELS GENERAUX

1- Introduction

Dans l'arsenal des outils informatiques offerts aux statisticiens, les logiciels généraux se situent à l'opposé du langage de programmation. Nous appelons logiciel général un programme ou un groupe de programmes permettant de mettre en oeuvre de façon relativement intégrée un ensemble relativement cohérent de fonctions de stockage et/ou de calcul.

Cohérence

Il existe des logiciels généraux spécialisés dans le traitement des séries temporelles par exemple TSP, d'autres dans le traitement des données d'enquête, par exemple LEDA, d'autres dans l'analyse statistique des données individuelles, par exemple BIBLOS, BMDP, SAS, d'autres dans l'étude des modèles économiques, par exemple SIMSYS, MODULECO, d'autres dans la manipulation de tableaux statistiques, par exemple ARGOSC.

Intégration

Ce terme recouvre les particularités de ces outils qui permettent d'éviter à l'utilisateur de passer son temps à programmer ses algorithmes, à programmer des enchaînements d'algorithmes déjà écrits, et dans les cas les plus favorables, à programmer des passages de données entre programmes distincts pour enchaîner des analyses différentes.

Il existe sur le marché un grand nombre de logiciels généraux susceptibles d'être utilisés par des statisticiens. Les logiciels généraux sont des programmes un peu plus complexes que la moyenne, mais des programmes quand même. Les difficultés qu'on a mentionnées dans les considérations sur l'écriture de programmes spécifiques, doivent être évoquées à leur propos.

D'autre part, ces programmes doivent permettre à tout statisticien de résoudre, relativement aisément, une gamme plus ou moins large de problèmes. Certains sont d'ailleurs énormément répandus et utilisés.

Enfin, ces programmes font souvent l'objet d'un commerce. Ce sont des

produits qui s'achètent ou se louent, sont théoriquement maintenus en bon état de marche, corrigés de leurs erreurs, et développés par leurs auteurs ou leurs distributeurs.

Tout ceci fait qu'il est sain d'éprouver à leur égard une certaine méfiance, et que les pratiques "consumeristes" d'exploitation systématique de leurs capacités et de tests comparatifs sont à envisager très sérieusement. Ce genre de préoccupations remonte au début des années 1970, surtout parmi la gent statisticienne anglosaxonne. Les publications où l'on peut en voir la trace sont par exemple :

- les comptes rendus des congrès annuels Interface (Etats-Unis),
- les comptes rendus des congrès annuels de la Section Calcul de l'association des Statisticiens américains,
- les comptes rendus des congrès bisannuels COMPSTAT (Europe),
- les journaux de la Société Royale de Statistique (Grande Bretagne) et surtout Applied statistics,
- le périodique Statistical Software Newsletter édité à Munich,
- la revue American Statistician.

L'AICSO, section de l'Institut International de Statistique pour le Calcul Statistique sur Ordinateur, a fourni depuis sa création, un gros effort dans ce domaine, et publié en 1979, sous l'autorité du Professeur FRANCIS, une étude comparative des logiciels statistiques, étude qui doit être actualisée en 1981.

A l'INSEE, ces travaux sont suivis à la Division Logiciels, dont une des missions est, en liaison avec les utilisateurs, le choix, l'installation et la diffusion des logiciels qui peuvent rendre service.

Il n'est pas facile d'évaluer correctement, encore moins de comparer, des logiciels statistiques. Des listes de critères ont été établies. Des tentatives de recensements des programmes existants ont été faites, qui présentent en général le point de vue des auteurs et sont donc peu critiques

L'obsolescence de ces papiers est normalement assez rapide, dans la mesure où par exemple les constructeurs de logiciels conscients et qui en ont les moyens corrigent normalement assez vite les défauts ponctuels repérés. L'enjeu

est important. De fait, une grande partie de la production de chiffres d'un organisme tel que l'INSEE repose sur l'emploi de ces logiciels. Que ces chiffres ne soient pas faux ne semble pas être une exigence démesurée. Il peut être souhaitable également qu'ils ne reviennent pas trop cher, ni en temps humain, ni en temps machine, qu'ils ne soient pas obtenus par des méthodes suspectes, etc .

2- Qualité d'un logiciel général

Les critères d'évaluation peuvent être classés de multiples façons. Nous utiliserons en gros la nomenclature de FRANCIS, HEIBERGER et VELLEMAN publiée dans American Statistician en février 1975.

a) Attrait pour l'utilisateur

Nous regroupons sous ce chapeau les caractéristiques qui peuvent donner à l'utilisateur éventuel l'envie d'employer un logiciel donné.

Qualité de la documentation

Ce point est fondamental. Les utilisateurs se plaignent souvent et à très juste titre de la mauvaise qualité de la documentation qui leur est proposée, que ce soit au plan de la présentation, du vocabulaire employé, de la langue (souvent anglaise, avant traduction éventuelle), etc .

Existence d'un langage de contrôle

Un langage de contrôle en format libre est préférable à un codage rigide. Il n'est a priori pas plus difficile à apprendre. Le problème peut venir du fait que les langages de contrôle les plus évolués, qui permettent les traitements les plus sophistiqués (enchaînements, répétitions, traitements alternatifs), finissent par ressembler de près à des langages de programmation, et donc nécessiter de la part des utilisateurs, un investissement non négligeable. Les langages de contrôle sont décodés par des programmes spéciaux appelés interpréteurs, traducteurs ou compilateurs. Les techniques de compilation actuellement connues permettent d'assurer un décodage rapide et d'effectuer une détection des erreurs efficace. Les logiciels, dont le langage de contrôle obéit aux règles de l'art, permettent une mise au point des demandes très rapides.

Types de données manipulables (structures de données)

Les logiciels statistiques actuellement disponibles permettent de manipuler un ou plusieurs des objets suivants : scalaire, vecteur (série temporelle), matrice, tableau observation X variable, tableau à plusieurs dimensions, etc ... Un logiciel tel que MODULECO manipulera des objets plus originaux, tels que : ensemble de séries, modèle ou matrice d'incidence.

Manipulation des données

Préalablement à toute analyse, le statisticien opère en général une mise en forme des données. Un logiciel général est d'autant plus agréable à utiliser qu'il permet des manipulations aisées des structures de données qu'il gère. Signalons le grand intérêt pour les statisticiens, des logiciels incorporant la notion de valeur manquante.

Interactivité

L'exposé complet de ce qu'on entend par là, exigerait des pages et des pages. Le logiciel interactif, au sens propre du terme, doit aider l'utilisateur à formuler sa demande, et fournir les réponses au terminal lorsque la demande est correcte.

Des logiciels tels que TSO, BIBLOS, ARGOSC, SAS ne sont interactifs : ils ne font que répondre passivement aux ordres donnés par l'utilisateur. A la réflexion et à l'expérience, on peut s'apercevoir qu'il est difficile de concevoir des logiciels interactifs entièrement satisfaisants pour tous les utilisateurs, débutants ou très avertis. Leur verbosité doit être réglabe. Ils doivent être assez souples pour permettre de différer les traitements lourds et diriger vers les imprimantes lourdes les sorties volumineuses. Actuellement, il existe encore peu de logiciels statistiques puissants interactifs vraiment très bien conçus, et leur coût d'emploi est souvent très élevé. Si cependant leur utilisation entraîne des réductions fortes de consommation de temps humain, leur développement est inévitable.

Sorties imprimées

Elles sont plus ou moins plaisantes. L'utilisateur peut avoir plus ou

moins de pouvoir sur leur apparence, leur détail, leur volume.

Sorties graphiques

Il existe actuellement des terminaux graphiques, tables traçantes ou écrans graphiques, en noir et blanc ou en couleur. Certains logiciels permettent des sorties sophistiquées sur de tels matériels.

Coût d'emploi

Ce n'est pas un critère très fondamental à l'INSEE. Il faudra sûrement finir par s'en préoccuper. D'une part, l'emploi de terminaux légers, d'autre part l'usage de logiciels généraux induisent inévitablement des sur-coûts par rapport à des programmes spécifiques bien faits et au traitement par lots classiques. Ils peuvent engendrer des gaspillages extrêmement importants, si les utilisateurs pensent remplacer les étapes de réflexion indispensables par des avalanches de calculs inutiles.

Domaine d'application

Les logiciels généraux visent en général un certain type d'applications -ils sont plus ou moins spécialisés- et un certain public. Ils peuvent exiger une compétence statistique variable.

b) Efficacité statistique

Souplesse

Les processus d'analyse statistique, que ce soit en analyse des données ou en modélisation, pour prendre deux exemples, sont essentiellement dynamiques, cheminant par une suite d'étapes non forcément prévues au départ.

La recherche d'une spécification nécessite de nombreux tâtonnements, une variante de modèle peut donner l'idée de dix variantes différentes, l'étude d'un tableau peut nécessiter des suppressions de points, des agrégations, des recodifications, etc .

Le logiciel vraiment intéressant est celui qui permet à l'utilisateur d'enchaîner ces différentes étapes sans douleur excessive (sans avoir par exemple à créer un nombre abusif de fichiers).

Précision

Un logiciel général doit être irréprochable sur ce plan, vu qu'il peut être mis à toutes les sauces, utilisé pour traiter des données de toutes origines, par des utilisateurs de tous calibres.

Etre irréprochable ne veut pas forcément dire fournir des résultats illusoirement précis. De nombreuses recherches des numériciens visent à prendre en compte dans les algorithmes la qualité des données d'entrée. Pour un certain nombre de techniques, encore trop rare, il est possible d'affirmer que tel problème est mal posé, ou que telles données sont mal conditionnées.

Mieux valent, en tout état de cause, des résultats mathématiquement justes, même illusoires, que des résultats faux.

Des considérations exposées dans un paragraphe précédent, le lecteur doit avoir retenu que ce dernier cas de figure peut se produire, malheureusement. Un article publié dans American Statistician en février 1977, a fait une certaine sensation, qui mettait en évidence le caractère totalement erroné de matrices de corrélation calculées à l'aide de logiciels extrêmement répandus, à partir de données pas spécialement bizarres et pas particulièrement volumineuses.

Extensibilité

Aucun logiciel, considéré à un instant donné, ne permettra la mise en oeuvre de toutes les techniques imaginées par le statisticien. On peut envisager de remédier à ce défaut inévitable de trois façons :

- la première solution est simple : il suffit que le logiciel défaillant autorise la sortie sur disque (ou sur bande) des données concernées, qui pourront ainsi être par exemple réexploitées au moyen d'un autre programme. Pour simple qu'elle soit, cette solution n'est que rarement possible. Les auteurs de logiciels aiment à confiner leurs utilisateurs dans le champ clos de leur création.

D'où l'équation :

Logiciels généraux = routine = mort de l'imagination.

- la deuxième solution est la suivante : le langage du logiciel est suffisamment puissant pour que le statisticien puisse programmer dans ce langage l'algorithme de son choix. Par exemple, si le langage offre des possibilités de calcul matriciel il pourra résoudre certains problèmes de statistique multidimensionnelle.

- la dernière solution est encore plus élaborée, et donc beaucoup plus rare que les précédentes. Elle consiste à autoriser l'utilisateur à ajouter sa méthode au logiciel, et ce au moindre coût, coût de programmation s'entend. Des recherches théoriques et pratiques sont menées à ce sujet, pour mettre en évidence les architectures de logiciels adéquates.

Portabilité

Nous retrouvons un problème déjà abordé. Les statisticiens de l'INSEE reliés à un IRIS 80 voudraient bien pouvoir disposer des mêmes outils que leurs collègues reliés à un IBM (et réciproquement, mais c'est le plus souvent comme ça que ça se passe, surtout en matière d'études, pour des raisons évidentes).

Le logiciel cent pour cent portable n'existe pas. Il y a toujours un petit quelque chose à changer pour l'adapter sur un nouveau matériel. Et pour atteindre un degré de portabilité raisonnable, les auteurs de logiciels doivent s'imposer des contraintes sévères, par exemple :

- quant au langage de programmation retenu, il faut le choisir parmi les langages les plus répandus et les plus standardisés (FORTRAN est un bon candidat, mais si on le choisit, il faut en plus se limiter au sous-ensemble des instructions FORTRAN les plus universelles) ;

- quant aux types de fichiers utilisés, tous les ordinateurs n'offrent pas la même gamme de types de fichiers (accès direct, séquentiel indexé, etc) ;

- quant à l'architecture du logiciel, etc .

Ils ne s'évitent cependant pas des coûts non nuls lors des adaptations successives à d'autres matériels.

Du côté des utilisateurs, le bilan peut également comporter des éléments négatifs. Le logiciel portable peut ne pas disposer de certaines caractéristiques agréables, simplement parce qu'il est impossible de les programmer de façon portable. Il peut également être d'usage coûteux, parce qu'il n'utilise pas au mieux les possibilités des ordinateurs sur lesquels il fonctionne.

Coût d'achat ou de location

La plupart des logiciels généraux d'origine universitaire bénéficiaient jusqu'à ces dernières années de tarifs très bas.

Dans un mouvement qui a pris naissance aux Etats-Unis, ces tarifs ont été fortement relevés, sans toutefois atteindre en général ceux pratiqués par les sociétés de services françaises ou par la compagnie IBM. Il y aurait en fait beaucoup à écrire sur la distribution de ces produits en France.

Arrêtons là l'exposé. D'autres points importants pourraient être développés. L'analyse faite précédemment mériterait d'être raffinée, précisée, nuancée.

3- Les relations entre les utilisateurs et les installateurs de logiciels

A l'INSEE, les relations sont pauvres et souvent à sens unique. La Division Logiciels conçoit ou choisit les logiciels à installer, fournit la documentation, organise les séances de formation. Il existe une assistance technique dont le rôle dans plus de 99 % des cas consiste à corriger les erreurs des utilisateurs.

Les erreurs de programme remontent difficilement. Au niveau de la documentation, on peut contempler une certaine grogne teintée de résignation. Les informaticiens sont jugés décidément incapables d'écrire en français et de façon

compréhensible. Ce reproche est peut-être justifié. Les utilisateurs pourraient cependant faire plus souvent part de leurs critiques et suggestions.

En ce qui concerne la formation, constatons que même si elle est organisée correctement, même si elle "passe bien", elle ne sert à rien si l'acquis n'est pas mis en pratique immédiatement. Rien ne s'oublie plus vite qu'un mode d'emploi informatique.

Au niveau des investissements eux-mêmes, il n'est pas facile de faire coopérer la Division Logiciels et les utilisateurs potentiels. Lors de la conception d'ARGOSC, des réunions furent organisées. Des logiciels tels que SAS ont été installés à l'essai, certains utilisateurs servant de cobayes. Il n'est guère facile d'aller plus loin, étant donné que les préoccupations des utilisateurs sont très diverses et que le Département de l'Informatique a sa stratégie propre.

Pour les logiciels très spécialisés de calcul, les utilisateurs peuvent cependant trouver toute assistance et documentation utiles à leur choix à la Division Logiciels.

La maintenance

On a fait allusion précédemment à ce problème épineux. Tout logiciel (programme) contient des erreurs, dont le nombre est fonction de sa taille et de sa complexité.

Installer un logiciel nouveau, susceptible d'être utilisé plusieurs centaines de fois par mois, c'est courir un risque multiforme : risque de se voir submergé sous les récriminations des utilisateurs si les erreurs détectées sont trop nombreuses, risque de faire publier des résultats faux en cas d'erreurs non détectées, etc .

La Division Logiciels doit minimiser ce risque, au moins pour les logiciels les plus courants. Ceux qui sont conçus à l'INSEE sont maintenus en permanence en bon état de marche. Pour ceux qui viennent de l'extérieur, un critère important de choix est l'existence d'une équipe de développement et de maintenance suffisamment nombreuse, compétente et motivée. Peu de logiciels statistiques satisfont correctement à ce critère.

En tout état de cause, la Division Logiciels essaie de se donner les moyens d'assurer la maintenance des produits qu'elle installe. Dans cette mesure, il est fondamental que les utilisateurs, qui découvrent des erreurs, se manifestent auprès des diverses cellules chargées de l'assistance informatique, afin que ces erreurs soient corrigées. La qualité de ces outils ne progressera qu'à cette condition.

Pourquoi utiliser des logiciels généraux ?

Un précédent chapitre rend compte des qualités dont devraient être pourvus des logiciels généraux bien faits. En premier lieu, il va de soi que certaines de ces qualités sont contradictoires. Ainsi la portabilité s'oppose souvent à la souplesse, à moins d'être recherchée seulement à l'intérieur d'une catégorie bien précisée de machines, par exemple de haut de gamme, offrant à l'intérieur de leur système d'exploitation standard des outils tels que : allocation dynamique de fichiers ou de mémoire. D'autre part, il faut bien reconnaître que les logiciels existants ne font pas tous preuve d'un grand nombre de ces qualités, quelle que puisse être la subjectivité d'un tel jugement.

Les considérations développées dans l'introduction permettent d'expliquer cet état de fait. Le coût de ces programmes est finalement élevé. Beaucoup de projets futuristes ont avorté faute de moyens (ou de suite dans les idées). Les dates de mises en chantier des logiciels américains d'analyse statistique sont étonnamment reculées -BMDP 1960- SAS 1966 - SPSS 1967.

Leur conception date ou leur architecture a été modifiée plusieurs fois, profondément parfois (ce qui n'a pas été sans poser des problèmes aux utilisateurs confrontés à des changements importants).

Le marché lui-même n'est pas gigantesque et ne se renouvelle pas très vite, vu la "routinisation" et le conservatisme importants (même s'il est justifié) des utilisateurs.

Malgré tout, les logiciels généraux rendent déjà depuis longtemps des services éminents de l'INSEE en particulier, DMS, METRIC eussent-ils été menés à bien en l'absence de SIMSYS, quels que soient ses défauts évidents ? Des dizaines d'enquêtes lourdes ont été tabulées à l'aide de LEDA, malgré ses imperfections.

Ces considérations banales, méritent d'être répétées sans cesse, si cela permet d'éviter que se renouvellent à chaque génération les mêmes erreements et que le statisticien frais émoulu de l'ENSAE veuille contrôler et tabuler son enquête en FORTRAN, au prix de gaspillages de temps et d'argent importants. L'avenir pour le "selfeur" comme on dit, horriblement, ce ne sont pas les outils archaïques d'il y a trente ans, mais des outils modernes, souples, raisonnablement faciles d'emploi, et parlant son langage, le problème fondamental étant en dernière analyse celui de leurs limites, problème actuellement mal perçu dans la majorité des cas, car dissimulé sous le mythe. Il est essentiel, pour l'INSEE, de suivre attentivement, sinon d'assurer, du moins à la mesure actuelle de ses moyens, le développement de tels outils.